Safe Execution of Temporally Flexible Plans for Bipedal Walking Devices

Andreas Hofmann and Brian Williams

Computer Science and AI Lab, Massachusetts Institute of Technology
Author(s) Address(es) Go(es) Here in 9 Point Times Roman
Author(s) Address(es) Go(es) Here in 9 Point Times Roman
hofma@csail.mit.edu, williams@mit.edu

Abstract

Plans with temporal flexibility have been used to allow discrete systems to adapt to disturbances that occur while the plan is being executed. To control more complex devices, such as bipedal walking machines, we must extend this execution paradigm to the control of hybrid (discrete/continuous) systems. Systems of this type are difficult to control for two reasons: 1) their high dimensionality and nonlinearity make control complex, even under nominal circumstances; and 2), operation of such systems in unstructured environments requires robustness to significant disturbances.

We introduce a novel approach to hybrid temporally flexible plan execution that achieves robustness by transforming the high-dimensional system into a set of low-dimensional weakly-coupled systems. This allows us to apply dynamic controllability concepts previously applied to discrete systems. We accomplish this decoupling using three components: 1) a feedback linearizing controller which provides basic decoupling, 2) a hybrid plan dispatcher which utilizes plan flexibility to adjust control settings for individual decoupled variables, and 3) plan compilation which computes bounds for the dispatcher's adjustments to control settings that satisfy plan requirements. We show the interaction of these components in control of a bipedal walking machine.

Introduction

Effective use of autonomous robots in unstructured, human environments requires that robots: 1) have sufficient autonomy to perform useful tasks independently, 2) have sufficient size, strength, and speed to accomplish such tasks in a timely manner, and 3) operate safely. A particularly challenging example of such a robot is a bipedal walking machine (Fig. 1a).

An example task for such a system is to walk to a soccer ball and kick it. If the system encounters a significant force disturbance while performing this task, it will have to compensate by changing its stepping pattern, or leaning the body as shown in Fig. 1b. The disturbance may cause a delay (allowing another player to kick the ball). At a more subtle level, it may interfere with movement synchronization; a lateral disturbance may cause synchronization problems with forward motion and stepping.

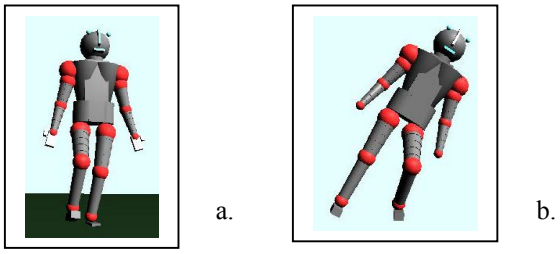

Fig. 1 – a. Biped, b. reaction to disturbance

This type of problem presents a number of technical challenges. First, movement task goals and safety requirements, which are most naturally specified in terms of state-space region, and temporal range constraints, must be translated into control actions that achieve these goals. This is difficult because the system is highly nonlinear, has high dimensionality, and has input constraints that limit controllability. Second, the dual goals of timely task execution and safety are often in conflict, and must be judiciously balanced. Finally, the system must be robust to significant disturbances.

Dynamic optimization techniques have been used to generate humanoid motion plans for animation applications [Popovic and Witkin, 1999]. However, these methods produce very detailed and inflexible reference trajectories, and are therefore not robust to disturbances. Robustness requires plan flexibility.

A powerful set of methods has been developed for discrete systems, for safe execution of temporally flexible plans [Morris, 2001]. These methods guarantee successful plan execution, as long as temporal uncertainty of activities is appropriately bounded. For example, in Fig. 2a, a car and sailboat leave Boston for P-Town at the same time. Duration of the sail is uncertain, but the uncertainty is bounded (between 6 and 12 hours). Likewise, the drive is between 3 and 4 hours. Synchronization in P-Town is assured because the car can wait there indefinitely for the sail boat.

In such systems, state is represented by logical variables. Constraints include logical constraints on these variables, and continuous temporal constraints. Executives for such

systems operate by scheduling start times of activities dynamically. They assume that at the end of an activity (like drive), state will not change (the car will remain in P-Town).

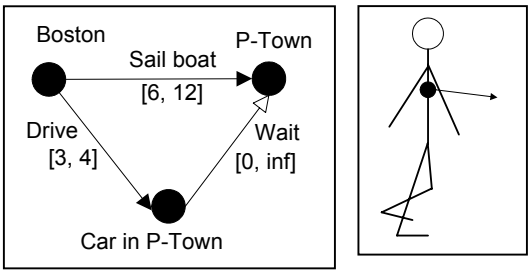

Fig. 2 – a. Dynamically controllable plan for discrete system (left), b. Underactuated dynamic system cannot always wait (right)

This approach is appropriate for many kinds of applications, but it does not work for agile underactuated dynamic systems like bipeds, where movement is fast and controllability is limited. In particular, the lack of equilibrium points in such systems means that state is constantly changing, and the executive cannot assume the system will wait in a particular state at the end of an activity. For example, in dynamic walking, it is not possible to instantly stop forward movement in the middle of a step, as shown in Fig. 2b. The stepping foot must move out in front, or the biped will fall.

Systems of this type include continuous, as well as discrete state variables, so we refer to such systems as *hybrid*. Continuous constraints on the continuous state variables express the system's dynamics, and specify valid regions of operation. An executive for such a system must take into account the system's dynamics and controllability limits. Rather than directly scheduling activity start times, the dispatcher controls timing indirectly. By adjusting control parameters appropriately, the dispatcher guides trajectories of interest into goal state-space regions at the right time.

Approach

We allow specification of task goals in terms of state-space region and temporal range constraints, as shown in Fig. 3. Foot placement constraints define qualitative poses such as double support, or left single support, but the details of the joint positions and trajectories are omitted. A state-space goal region for the forward position of the center of mass at the end of the gait sequence defines the task goal. A temporal range constraint specifies task completion time requirements.

To translate these specifications into control actions, we use a mixed on-line/off-line approach. The off-line component is a compiler that synthesizes a set of adaptive controllers. The on-line component is a hybrid dispatcher that efficiently adapts control settings in response to disturbances.

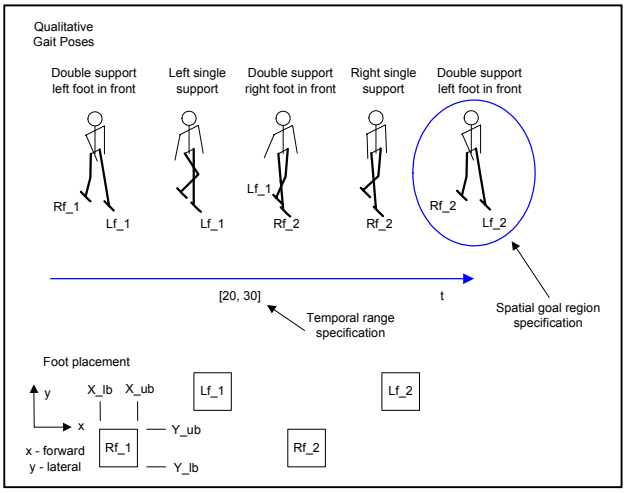

Fig. 3 – Task Specification

A key insight in our approach is to use a feedback linearizing multivariable controller [Hofmann, 2004] to transform the highly nonlinear, tightly coupled biped system into a loosely coupled set of linear 2nd-order single-input single-output (SISO) systems. This *SISO abstraction* greatly simplifies the task of the hybrid dispatcher, allowing it to focus on a few quantities of interest (center of mass position), which appear to behave in an independent and linear manner, rather than on the details of joint movement. In particular, the SISO abstraction allows the dispatcher to control the system by adjusting a small set of linear control law parameters for each quantity of interest. The multivariable controller then takes care of the details of computing torques that achieve the desired motion.

Qualitative State Plan and Plant

State-space and temporal constraints that specify goal regions are assembled into a *qualitative state plan*, which represents the desired state evolution of the *plant* (the system being controlled). The state plan is qualitative in that it specifies behavior in terms of state regions with common characteristics, rather than with specific states. The problem, given a state plan and a plant, is to generate a sequence of control actions that move the plant to a state consistent with that required by the plan, while avoiding unsafe regions.

Plant

The plant is represented by a set of dynamic equations that specify state evolution as a function of inputs. We define a plant by the tuple $\langle \mathbf{x}, \mathbf{y}, \mathbf{u}, \mathbf{h}, \mathbf{f} \rangle$ where $\mathbf{x}, \mathbf{y}, \mathbf{u}$ is the set of state, output (controlled), and input variables, respectively,

$\mathbf{h}$ is a set of algebraic equations that relate outputs to inputs and state, and $\mathbf{f}$ is a set of 1st-order differential equations that describe state evolution. The input vector, $\mathbf{u}$, includes control inputs, such as joint torques, and also environment inputs (ground contact forces). The state vector, $\mathbf{x}$, includes continuous variables, such as joint positions and velocities, and discrete variables, to indicate the presence of environment forces. The output vector, $\mathbf{y}$, contains the variables to be controlled, such as forward and lateral center of mass (CM) position and velocity. Our simulated walking biped plant has 18 degrees of freedom, and is highly nonlinear [Hofmann et al., 2002]. We assume that plant state is available from sensors, or can easily be estimated.

Qualitative State Plan

A qualitative state plan specifies state evolution using sequences of *activities* as shown in Fig. N. Each sequence (each row in this diagram) specifies behavior for a particular quantity. Two types of quantities, *controlled quantities*, and *input quantities*, can be specified. Controlled quantities are position/velocity pairs corresponding to elements of $\mathbf{y}$. Input quantities are scalar functions, $g_i(y_i, \dot{y}_i)$, that represent control laws for controlled quantities, and thus, correspond to elements of $\mathbf{u}$. In Fig. 4, forward and lateral CM are examples of controlled quantities, and forward and lateral CP (center of pressure) are examples of input quantities.

Each activity in a sequence is part of a control epoch (column in Fig. 4). The control epochs shown in Fig. 4 correspond to the qualitative poses shown in Fig. 3. Thus, epoch 1 represents double support with the left foot in front, epoch 2, left single support, epoch 3, double support with the right foot in front, and epoch 4, right single support. Epoch 5 repeats epoch 1, but is one gait cycle forward. Vertical bars in Fig. 4 between rows represent synchronization constraints, so that all quantities advance to the next control epoch at the same time.

Each activity may have a temporal duration range constraint, indicated by [lb, ub]. This specifies the lower bound (lb) and upper bound (ub) on the activity's duration. In addition, range constraints on acceptable initial and goal regions for quantities may be specified. For controlled quantities, regions are specified using rectangles in position/velocity phase space. For input quantities, regions are specified using scalar ranges. Note that the duration and region constraints are optional. In fact, these constraints are omitted for many of the activities in the state plan. In Fig. 4, we care about the initial and final region of the CM, but not about the details in between. Thus, for the CM quantities, an initial region for epoch 1 may be specified, and a final region for epoch 5, but the intervening regions may be left unspecified. Similarly, we care that the gait cycle be completed within time range [t_lb, t_ub], but not about the detailed durations of each activity, so these are unspecified as well.

Each activity may also have a *tube* constraint specifying a required region for the associated quantity over the entire duration of the activity. Such constraints are useful, for example, for limiting the range of forward and lateral CP, over the entire course of a qualitative pose. This is important because CP is an input quantity that is limited by foot placement (see also Fig. N-1). For example, in epoch 2 (left single support), the CP is restricted to the support region under the left foot.

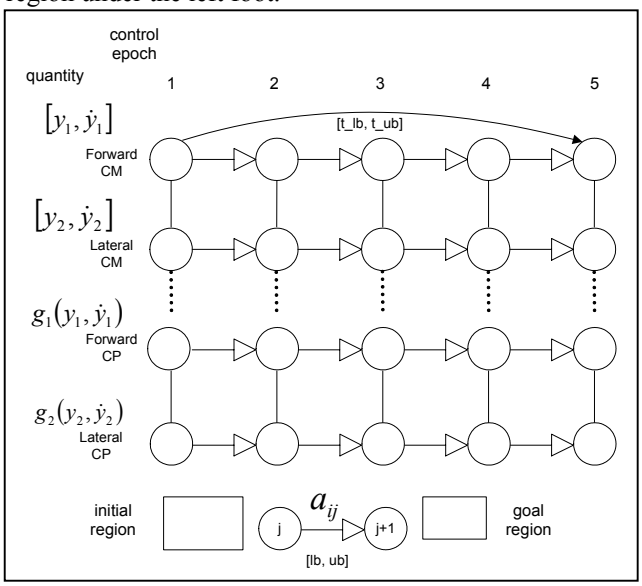

Fig. 4 – Qualitative State Plan

Formally, we define a state plan as a set, A, of activities, $a(i, j)$, where i refers to the quantity, and j to the control epoch. An activity is defined by the tuple $\langle R_{init}, R_{tube}, R_{goal}, R_{temporal}, a_{next} \rangle$ where R_{init}, R_{tube}, and R_{goal} specify, respectively, initial, operating, and goal regions in state space for the controlled variable associated with the activity, and $R_{temporal}$ specifies temporal constraints. The activity a_{next} is the activity to transition to when the current activity is finished.

The region constraints, R_{init}, R_{tube}, and R_{goal} are of the form $(y_{i\min} \le y_i \le y_{i\max}) \wedge (\dot{y}_{i\min} \le \dot{y}_i \le \dot{y}_{i\max})$ for controlled quantities, and $(y_{i\min} \le y_i \le y_{i\max})$ for input quantities. Use of such unary constraints with constant bounds results in rectangular regions in position/velocity state-space. This implies more conservative bounds than would be possible if the constraints were multivariable and nonlinear, but it also greatly simplifies planning. An activity may begin if its quantity is in the region defined by R_{init}. An activity cannot end unless the quantity is in R_{goal}. The quantity must stay within R_{tube} for the entire duration of the activity.

$R_{temporal}$ is of the form $\langle R_{duration}, A_{parallel} \rangle$, where $R_{duration}$ is a simple temporal constraint $[lb, ub]$ specifying the permissible range of activity duration, and $A_{parallel}$ is a set of activities that must finish simultaneously with the current one. $A_{parallel}$ provides the capability to synchronize multiple concurrent activity sequences (vertical bars in Fig. N). For example, for a biped, movement of the stepping foot must be synchronized with forward movement of the center of mass.

An activity finishes if its R_{goal}, $R_{duration}$, and $A_{parallel}$ constraints are satisfied. After it finishes, it transitions to the successor, a_{next}, immediately. An activity, a, is executed successfully iff there exists a start time, ts, and a finish time, tf, for the activity, such that $lb \le tf - ts \le ub$, and there exists a trajectory for the associated controlled variable y such that $y(ts), \dot{y}(ts)$ satisfy R_{init}, $y(tf), \dot{y}(tf)$ satisfy R_{goal}, and $y(t), \dot{y}(t)$ satisfy R_{tube} for $ts \le t \le tf$. A state plan is executed successfully iff each activity, $a(i,j)$, is executed successfully, and the associated finish time, $tf(i,j)$, is such that if the activity has a successor, $a(i,j+1)$, then $tf(i,j) = ts(i,j+1)$, and for any parallel activity, $a(k,j)$, listed in $A_{parallel}$, $tf(i,j) = tf(k,j)$. Fig. 5 shows lateral CM and CP trajectories for a nominal execution of the state plan shown in Fig. 4.

Plan Compiler

A qualitative state plan cannot be executed directly because it is missing control information, and because much of the trajectory information is still under-specified. The plan compiler adds this missing information to the qualitative state plan, producing a qualitative control plan. A qualitative control plan consists of the activities from the state plan, with the two following additions: 1) control information is included, and 2) the region constraints, R_{init}, R_{tube}, and R_{goal}, and the duration constraint, $R_{duration}$, specify non-infinite bounds. Control parameter information is of the form $\langle k_{1\min}, k_{1\max}, k_{2\min}, k_{2\max} \rangle$; bounds on control parameters for the linear control laws used in the SISO abstraction. Thus, the qualitative control plan contains all the information needed to control the SISO system, and to monitor its status with respect to region and temporal bounds.

In computing bounds on the control parameters, the compiler is, in effect, performing an adaptive controller synthesis. The hybrid dispatcher utilizes the flexibility of the control parameter ranges to adapt control settings as needed. In order to maximize robustness to disturbances, the compiler attempts to maximize the size of initial regions, and tubes, minimize the size of goal regions, and maximize controllable temporal activity duration ranges. Maximizing initial regions and minimizing goal regions results in a contraction; the family of trajectories in the tube "contract" to each other as time advances. Maximizing controllable temporal range makes synchronization with other controlled quantities easier.

In generating trajectories, the compiler must take into account dynamics. We want fast performance, but due to the underactuated nature of the system, this leads to a reduction in controllability. Thus, future consequences of current actions become increasingly important as speed of movement is increased. The compiler must take into account future epochs in order to plan feasible trajectories. In this respect, the plan compiler is similar to receding horizon model-predictive controllers [ref. From Thomas' paper].

The compiler proceeds in two steps. First, it computes a nominal trajectory that reaches the goal state from the initial state, and that satisfies all state-space and temporal constraints. Second, to provide robustness to disturbances, the compiler expands the nominal trajectory, creating regions and tubes not specified explicitly in the qualitative state plan, attempting to maximize initial regions, minimize goal regions, and maximize controllable temporal range. For both steps, we utilize an SQP (Sequential Quadratic Programming) optimizer, and formulate the problem as an NLP (Nonlinear Program). These two steps are now described in detail.

Nominal Trajectory Computation

The NLP formulation for the nominal trajectory computation is as follows. For each activity, a_{ij}, associated with a controlled quantity, parameters to optimize are $\langle y_{init}, \dot{y}_{init}, y_{goal}, \dot{y}_{goal}, t_{goal}, k_1, k_2 \rangle$, where $y_{init}, \dot{y}_{init}$ are the initial state of the quantity associated with the activity, $y_{goal}, \dot{y}_{goal}$ are the quantity's final state, t_{goal} is the duration of the activity, and k_1, k_2 are parameters for the linear control law that achieves the trajectory. Constraints on these parameters are as follows. The trajectory is defined by the analytic solution to the linear 2nd-order differential equation formed by applying the linear control law to the SISO abstraction. This yields an equality constraint that relates goal to initial state, control parameters, and duration:

$$\begin{aligned} y_{goal} &= f_1(y_{init}, \dot{y}_{init}, k_1, k_2, t_{goal}) \\ \dot{y}_{goal} &= f_2(y_{init}, \dot{y}_{init}, k_1, k_2, t_{goal}) \end{aligned} \quad (1)$$

Continuity from one epoch to the next is expressed as

$$\begin{aligned} y_{goal}(i,j) &= y_{init}(i,j+1) \\ \dot{y}_{goal}(i,j) &= \dot{y}_{init}(i,j+1) \end{aligned} \quad (2)$$

Inequality constraints for R_{init}, R_{tube}, R_{goal}, and $R_{duration}$ are as described previously. Synchronization constraints across quantities are expressed as

$$\forall i,j : t_{goal}(i,j) = t_{goal}(i+1,j) \quad (3)$$

Finally, the temporal constraint on overall state plan execution time is given by

$$\forall i,j : t_{lb} \le \sum_i t_{goal}(i,j) \le t_{ub} \quad (4)$$

The cost function includes terms that maximize the distance to the R_{tube} boundaries.

An example of a nominal trajectory computed in this way is shown in Fig. 5.

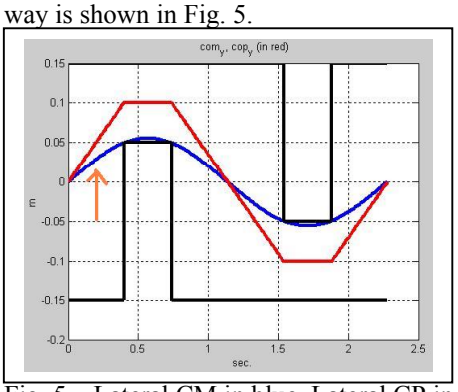

Fig. 5 – Lateral CM in blue, Lateral CP in red

Qualitative Control Plan Computation

The NLP formulation for the control plan computation is as follows. For each activity, a_{ij}, associated with a controlled quantity, parameters to optimize are: $\langle y_{init\,\min}, \dot{y}_{init\,\min}, y_{init\,\max}, \dot{y}_{init\,\max} \rangle$ (parameters of R_{init}), $\langle y_{goal\,\min}, y_{goal\,\min}, y_{goal\,\max}, y_{goal\,\max} \rangle$ (parameters of R_{goal}), $\langle t_{\min}, t_{\max} \rangle$ (parameters of $R_{duration}$), and $\langle k_{1\min}, k_{1\max}, k_{2\min}, k_{2\max} \rangle$, the bounds on the control parameters. Note that these are similar to the ones in the nominal computation, except that they are now ranges rather than nominal values.

In order to understand how this computation works, it is necessary to understand two trajectories that represent extremes of behavior: the *guaranteed fastest trajectory* (GFT), and the *guaranteed slowest trajectory* (GST). The GFT represents a lower bound on the time needed to get from anywhere in the initial region, to somewhere in the goal region. The GST represents the corresponding upper bound. For both these trajectories, it is assumed that velocity does not change sign (position is monotonically increasing or decreasing).

Consider the initial and goal regions shown in Fig. 6. For the GFT, the worst-case starting point in the initial region is point B, which corresponds to $y_{init\,\min}, \dot{y}_{init\,\min}$. This represents the slowest possible start. By accelerating as quickly as possible, the GFT reaches point D in the goal region, which corresponds to $y_{goal\,\min}, \dot{y}_{goal\,\max}$. This represents the fastest finish point in the goal region. For the GST, the worst-case starting point in the initial region is point A, which corresponds to $y_{init\,\max}, \dot{y}_{init\,\max}$. This represents the fastest possible start. By accelerating as slowly as possible, the GST reaches p;oint c in the goal region, which corresponds to $y_{goal\,\max}, \dot{y}_{goal\,\min}$. This represents the slowest finish point.

The times for each trajectory are designated t_{GFT} and t_{GST}. If $t_{GFT} < t_{GST}$, then there exists a temporal range, $[t_{GFT}, t_{GST}]$, during which the endpoint of a trajectory

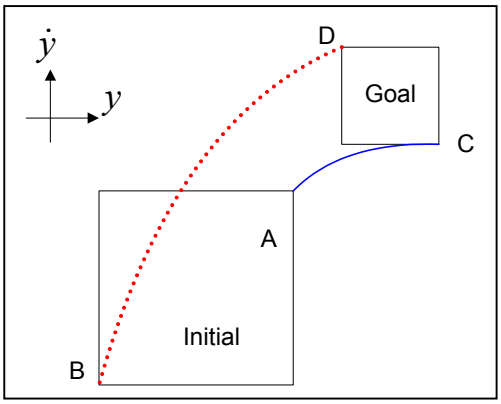

Fig. 6 – GFT (dotted) and GST (solid)

beginning from anywhere in the initial region can be guaranteed to be in the goal region. The existence of the temporal range is important for synchronizing with other controlled quantities. Thus, the GFT and GST are useful for determining a maximum initial region, given a particular goal region, such that the controllable temporal range exists.

GFT and GST can be understood intuitively by considering a very simple control law. Suppose that the only control input allowed is a single acceleration spike (of an appropriate size). If this spike is applied at the beginning of the trajectory, then maximum velocity is reached immediately. This corresponds to the GFT, as shown in Fig. 7. If this spike is applied at the end, then the trajectory will progress at minimum velocity until the end. This corresponds to the GST.

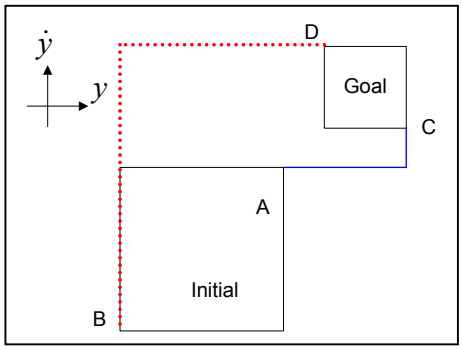

Fig. 7 - GFT (dotted) and GST (solid) for acceleration spike control laws

In the NLP formulation, existence of initial and goal regions is expressed as

$$\begin{aligned} y_{init\,\min} &< y_{init\,\max} \\ \dot{y}_{init\,\min} &< \dot{y}_{init\,\max} \\ y_{goal\,\min} &< y_{goal\,\max} \\ \dot{y}_{goal\,\min} &< \dot{y}_{goal\,\max} \end{aligned} \quad (5)$$

To guarantee contraction from one control epoch to the next, the goal region of an activity must fit inside the initial region of its successor.

$$\begin{aligned} y_{goal\,\min}(i,j) &\geq y_{init\,\min}(i,j+1) \\ \dot{y}_{goal\,\min}(i,j) &\geq \dot{y}_{init\,\min}(i,j+1) \\ y_{goal\,\max}(i,j) &\leq y_{init\,\max}(i,j+1) \\ \dot{y}_{goal\,\max}(i,j) &\leq \dot{y}_{init\,\max}(i,j+1) \end{aligned} \quad (6)$$

Constraints representing the GFT are expressed as

$$\begin{aligned} y_{goal\,\min} &= f_1(y_{init\,\min}, \dot{y}_{init\,\min}, k_{1\max}, k_{2\max}, t_{\min}) \\ \dot{y}_{goal\,\max} &= f_2(y_{init\,\min}, \dot{y}_{init\,\min}, k_{1\max}, k_{2\max}, t_{\min}) \end{aligned} \quad (7)$$

Constraints representing the GST are expressed as

$$\begin{aligned} y_{goal\,\max} &= f_1(y_{init\,\max}, \dot{y}_{init\,\max}, k_{1\min}, k_{2\min}, t_{\max}) \\ \dot{y}_{goal\,\max} &= f_2(y_{init\,\max}, \dot{y}_{init\,\max}, k_{1\min}, k_{2\min}, t_{\max}) \end{aligned} \quad (8)$$

The requirement for temporal controllability is expressed as $t_{\min} < t_{\max}$. Synchronization constraints are

$$\begin{aligned} &(t_{\min}(1,j) \leq t_{trans\,\min}(j) \leq t_{trans\,\max}(j) \leq t_{\max}(1,j)) \wedge \ldots \\ &(t_{\min}(i,j) \leq t_{trans\,\min}(j) \leq t_{trans\,\max}(j) \leq t_{\max}(i,j)) \end{aligned}$$

Thus, $[t_{trans\,\min}(j), t_{trans\,\max}(j)]$ is the temporal range when transition out of control epoch j may occur.

The cost function maximizes initial region size, minimizes goal region size, and maximizes controllable temporal range.

Fig. 8 shows regions for lateral CM computed in this way.

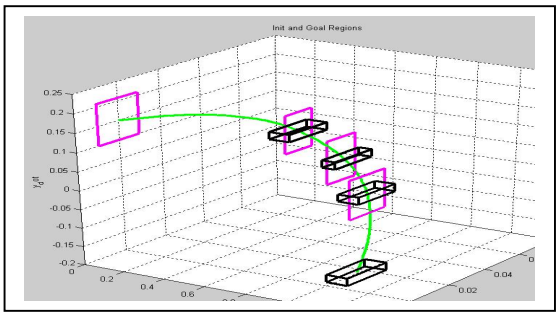

Fig. 8 – Lateral CM regions ($y, \dot{y}$ vs. time)

Temporal Uncertainty

If controllability is very limited, it may be difficult to achieve a large enough initial region. This situation can be improved by relaxing the constraint $t_{GFT} < t_{GST}$, as shown in Fig. 9.

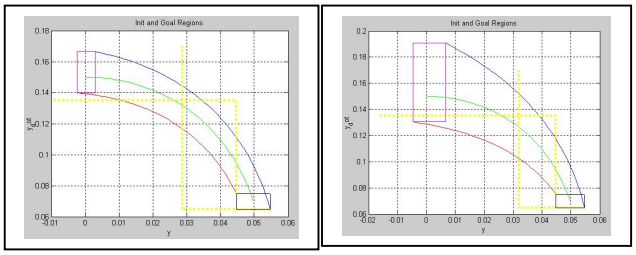

Fig. 9 – a. t(GFT) = t(GST), left, b. t(GFT) > t(GST), right

The intuitive explanation for why this happens is that as GFT is allowed to take longer, the B point of the initial region is allowed to stretch to the left (position decreases). Similarly, if GST is allowed to complete faster, point A moves to the right (position increases).

Unfortunately, relaxing this constraint means that we lose temporal controllability; the time of arrival in the goal region can no longer be precisely controlled. However, the uncertainty on arrival time is bounded by $[t_{GST}, t_{GFT}]$.

This may still be ok if controllability of other controlled quantities is strong enough to compensate for uncertainty. When this is the case, the system can be considered to be dynamically controllable [Morris, 2001]. This situation can be represented using an STNU (simple temporal network with uncertainty), as shown in Fig. 10.

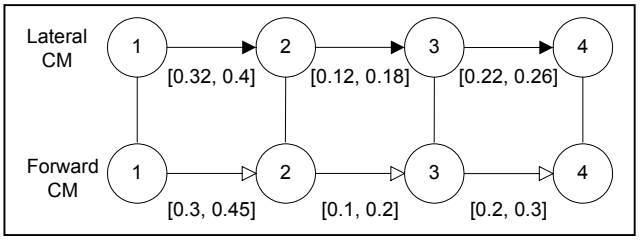

Fig. 10 – STNU for lateral and forward CM

In this STNU, the temporal durations for lateral CM are uncertain but bounded. The temporal durations for forward CM are certain, and they are large enough to compensate for the uncertainty in the lateral CM. Thus, transition synchronization can still be guaranteed, though this will require adjustment by the hybrid dispatcher once the uncertainty is resolved.

Hybrid Dispatcher

The hybrid dispatcher executes the qualitative control plan. It acts as an adaptive controller, adjusting control parameters within the limits specified in the control plan, to guide each controlled quantity into its goal region at the correct time. When all quantities are in their respective goal regions, the hybrid dispatcher transitions to the next control epoch. Note that unlike dispatchers for discrete systems [Morris, 2001], the hybrid dispatcher is not able to directly schedule start times for activities. Rather, it indirectly controls timing by adjusting control parameters.

At the beginning of a new control epoch, j, the dispatcher computes a target transition time, $t_{trans}(j)$. When there is no temporal uncertainty, it chooses a time in the range $[t_{trans\min}(j), t_{trans\max}(j)]$. When there is uncertainty in one of the controlled quantities, the transition time is determined when the uncertainty for this controlled quantity is resolved.

For each controlled quantity, the dispatcher then monitors progress by computing a prediction of the point in state space for the controlled quantity at $t_{trans}(j)$. This prediction is computed analytically in the same manner as eq. 1, so it is fast. If the predicted point is within the goal region, then the dispatcher does nothing. If it is outside the goal region, then the dispatcher adjusts the control parameters to attempt to move the predicted point back into the goal region. If this is unsuccessful, the plan execution fails, and the dispatcher requests a new plan. If all trajectories execute successfully, then when all controlled quantities are in their respective goal regions, the dispatcher transitions to the next control epoch.

Results and Discussion

Our tests on the simulated biped show that a small lateral CM disturbance (100 N for 0.01 sec) can be handled by the multivariable controller without any immediate action by the dispatcher. A larger (250 N) disturbance requires gain adjustment by the dispatcher. A disturbance of 300 N is too large for the dispatcher to handle by itself; a new dispatchable plan is required.

These tests demonstrate compliance to disturbances at three levels: 1) use of low-gain control, 2) flexibility in the dispatchable state plan allowing for adjustments by the hybrid dispatcher, and 3) fast reactive planning. This allows for integrated handling of disturbances of varying degrees of severity.

References

[Bradley and Zhao, 1993] E. Bradley and F. Zhao. Phase-space control system design. *Control Systems*, 13(2),39-46 April, 1993

[Goswami, 1999] A. Goswami. Postural stability of biped robots and the foot rotation indicator (FRI) point. *International Journal of Robotics Research*, July/August 1999

[Hofmann et al., 2002] A. Hofmann, M. Popovic, H. Herr. Humanoid Standing Control: Learning from Human Demonstration. *Journal of Automatic Control*, 12(1), 16–22

[Hofmann et al., 2004] A. Hofmann, S. Massaquoi, M. Popovic, and H. Herr. A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs. *Proc. International Conference on Intelligent Robots and Systems* (IROS). Sendai, Japan

[Morris et al., 2001] P. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal uncertainty. *Proceedings of the 17th International Joint Conference on A.I.* (IJCAI-01). Seattle (WA, USA).

[Muscettola et al., 1998] N. Muscettola, P. Morris, and I. Tsamardinos. Reformulating temporal plans for efficient execution. *Proc. Of Sixth Int. Conf. On Principles of Knowledge Representation and Reasoning*, 1998

[Popovic et al., 1999] Z. Popovic and A. Witkin. Physically based motion transformation. Siggraph 1999

[Popovic et al., 2004] M. Popovic, A. Hofmann, H. Herr. Zero spin angular momentum control: definition and applicability. (Humanoids). Los Angeles (CA, USA).

[Slotine and Li, 1991] J. Slotine and W. Li. Applied Nonlinear Control. Ch. 6, Prentice Hall, NJ, USA

[Williams and Nayak, 1997] B. Williams and P. Nayak. A Reactive Planner for a Model-based Executive. *Proceedings of the International Joint Conference on Artificial Intelligence* (IJCAI, 1997)